Introduction to Adaptive Filters

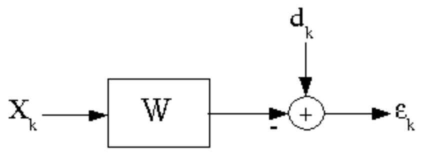
In many applications requiring filtering, the necessary frequency response may not be known beforehand, or it may vary with time. (Example; suppression of engine harmonics in a car stereo.) In such applications, an adaptive filter which can automatically design itself and which can track system variations in time is extremely useful. Adaptive filters are used extensively in a wide variety of applications, particularly in telecommunications.

Outline of adaptive filter material

1. **Wiener Filters** L optimal (FIR) filter design in a statistical context
2. **LMS algorithm** simplest and by-far-the-most-commonly-used adaptive filter algorithm
3. **Stability and performance of the LMS algorithm** When and how well it works
4. **Applications of adaptive filters** Overview of important applications
5. **Introduction to advanced adaptive filter algorithms** Techniques for special situations or faster convergence

Discrete-Time, Causal Wiener Filter

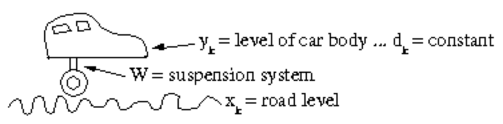
Stochastic L_2 optimal (least squares) FIR filter design problem: Given a wide-sense stationary (WSS) input signal x_k and desired signal d_k (WSS ⇔ $E[y_k] = E[y_{k+d}]$, $r_{\mathrm{yz}}(l) = E[y_k z_{k+l}]$, $\forall k, l : (r_{\mathrm{yy}}(0) < \infty)$)



The Wiener filter is the linear, time-invariant filter minimizing $E\left[\varepsilon^2\right]$, the variance of the error.

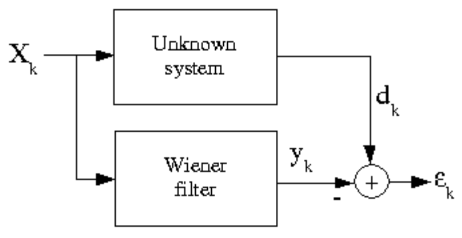
As posed, this problem seems slightly silly, since d_k is already available! However, this idea is useful in a wide cariety of applications.

Example:
active suspension system design



Note:optimal system may change with different road conditions or mass in car, so an **adaptive** system might be desirable.

Example:
System identification (radar, non-destructive testing, adaptive control systems)



Exercise:

Problem:

Usually one desires that the input signal x_k be "persistently exciting," which, among other things, implies non-zero energy in all frequency bands. Why is this desirable?

Determining the optimal length-N causal FIR Weiner filter

Note: for convenience, we will analyze only the causal, real-data case; extensions are straightforward.

$$y_k = \sum_{l=0}^{M-1} w_l x_{k-l}$$

$$\underset{w_l}{\operatorname{argmin}} E\left[\varepsilon^2\right] = E\left[(d_k - y_k)^2\right] = E\left[\left(d_k - \sum_{l=0}^{M-1} w_l x_{k-l}\right)^2\right] = E\left[{d_k}^2\right] - 2\sum_{l=0}^{M-1} w_l E[d_k x_{k-l}] + \sum_{l=0}^{M-1}\sum_{m=0}^{M-1} (w_l$$

$$E\left[\varepsilon^2\right] = r_{\mathrm{dd}}(0) - 2\sum_{l=0}^{M-1} w_l r_{\mathrm{dx}}(l) + \sum_{l=0}^{M-1}\sum_{m=0}^{M-1} w_l w_m r_{\mathrm{xx}}(l-m)$$

where

$$r_{\mathrm{dd}}(0) = E\left[{d_k}^2\right]$$

$$r_{\mathrm{dx}}(l) = E[d_k X_{k-l}]$$

$$r_{\mathrm{xx}}(l-m) = E[x_k x_{k+l-m}]$$

This can be written in matrix form as

$$E\left[\varepsilon^2\right] = r_{\mathrm{dd}}(0) - 2\boldsymbol{P}\boldsymbol{W}^T + \boldsymbol{W}^T R \boldsymbol{W}$$

where

$$\boldsymbol{P} = \begin{pmatrix} r_{\mathrm{dx}}(0) \\ r_{\mathrm{dx}}(1) \\ \vdots \\ r_{\mathrm{dx}}(M-1) \end{pmatrix}$$

$$R = \begin{pmatrix} r_{\mathrm{xx}}(0) & r_{\mathrm{xx}}(1) & \dots & \dots & r_{\mathrm{xx}}(M-1) \\ r_{\mathrm{xx}}(1) & r_{\mathrm{xx}}(0) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{\mathrm{xx}}(0) & r_{\mathrm{xx}}(1) \\ r_{\mathrm{xx}}(M-1) & \dots & \dots & r_{\mathrm{xx}}(1) & r_{\mathrm{xx}}(0) \end{pmatrix}$$

To solve for the optimum filter, compute the gradient with respect to the top weights vector W

$$\nabla \doteq \begin{pmatrix} \frac{\partial \varepsilon^2}{\partial w_0} \\ \frac{\partial \varepsilon^2}{\partial w_1} \\ \vdots \\ \frac{\partial \varepsilon^2}{\partial w_{M-1}} \end{pmatrix}$$

$$\nabla = -(2\boldsymbol{P}) + 2R\boldsymbol{W}$$

(recall $\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{W}}\left(\boldsymbol{A}^T W\right) = \boldsymbol{A}^T$, $\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{W}}(\boldsymbol{W}M\boldsymbol{W}) = 2M\boldsymbol{W}$ for symmetric M) setting the gradient equal to zero ⇒

$$W_{\text{opt}}R = \boldsymbol{P} \Rightarrow W_{\text{opt}} = R^{-1}\boldsymbol{P}$$

Since R is a correlation matrix, it must be non-negative definite, so this is a minimizer. For R positive definite, the minimizer is unique.

Practical Issues in Wiener Filter Implementation

The weiner-filter, $W_{\text{opt}} = R^{-1}\boldsymbol{P}$, is ideal for many applications. But several issues must be addressed to use it in practice.

Exercise:

Problem:

In practice one usually won't know exactly the statistics of x_k and d_k (i.e. R and $\boldsymbol{P}$) needed to compute the Weiner filter.

How do we surmount this problem?

Solution:

Estimate the statistics

$$\widehat{r_{\text{xx}}(l)} \simeq \frac{1}{N}\sum_{k\ \ 0}^{N-1} x_k x_{k+l}$$

$$\widehat{r_{\text{xd}}(l)} \simeq \frac{1}{N}\sum_{k\ \ 0}^{N-1} d_k x_{k-l}$$

then solve $\widehat{W_{\text{opt}}} = \widehat{R^{-1}} = P$

Exercise:

Problem:

In many applications, the statistics of x_k, d_k vary slowly with time.

How does one develop an **adaptive** system which tracks these changes over time to keep the system near optimal at all times?

Solution:

Use short-time windowed estiamtes of the correlation functions.

Note:

$$\left(\widehat{r_{\mathrm{xx}}(l)}\right)^{k} = \frac{1}{N}\sum_{m\quad 0}^{N-1} x_{k-m}x_{k-m-l}$$

$$\left(\widehat{r_{\mathrm{dx}}(l)}\right)^{k} = \frac{1}{N}\sum_{m\quad 0}^{N-1} x_{k-m-l}d_{k-m}$$

and ${W_{\mathrm{opt}}}^{k} \simeq \left(R_k\right)^{-1}P_k$

Exercise:

Problem: How can $\widehat{r_{\mathrm{xx}}^{k}(l)}$ be computed efficiently?

Solution:

Recursively!

$$r_{\mathrm{xx}}^{k}(l) = r_{\mathrm{xx}}^{k-1}(l) + x_k x_{k-l} - x_{k-N}x_{k-N-l}$$

This is critically stable, so people usually do

$$(1-\alpha)\;\; {r_{\mathrm{xx}}}^{k}(l) = \alpha r_{\mathrm{xx}}^{k-1}(l) + x_k x_{k-l}$$

Exercise:

Problem: how does one choose N?

Tradeoffs

Larger N → more accurate estimates of the correlation values → better $\widehat{W}_{\text{opt}}$. However, larger N leads to slower adaptation.

Note: The success of adaptive systems depends on x, d being roughly stationary over at least N samples, $N > M$. That is, all adaptive filtering algorithms require that the underlying system varies slowly with respect to the sampling rate and the filter length (although they can tolerate occasional step discontinuities in the underlying system).

Computational Considerations

As presented here, an adaptive filter requires computing a matrix inverse at each sample. Actually, since the matrix R is Toeplitz, the linear system of equations can be sovled with $O\ M^2$ computations using Levinson's algorithm, where M is the filter length. However, in many applications this may be too expensive, especially since computing the filter output itself requires $O(M)$ computations. There are two main approaches to resolving the computation problem

1. Take advantage of the fact that R^{k+1} is only slightly changed from R^k to reduce the computation to $O(M)$; these algorithms are called Fast Recursive Least Squareds algorithms; all methods proposed so far have stability problems and are dangerous to use.
2. Find a different approach to solving the optimization problem that doesn't require explicit inversion of the correlation matrix.

Note: Adaptive algorithms involving the correlation matrix are called **Recursive least Squares** (RLS) algorithms. Historically, they were developed after the LMS algorithm, which is the slimplest and most widely used approach $O(M)$. $O\ M^2$ RLS algorithms are used in applications requiring very fast adaptation.

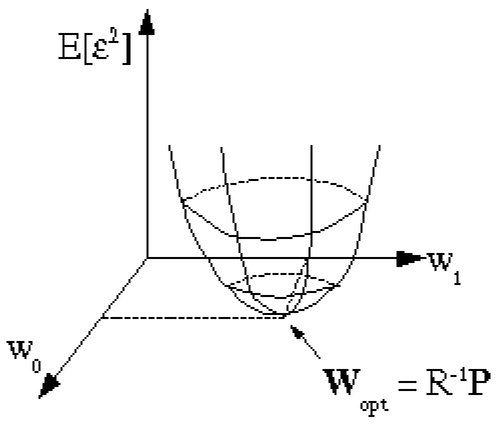
Quadratic Minimization and Gradient Descent

Quadratic minimization problems

The least squares optimal filter design problem is quadratic in the filter coefficients:

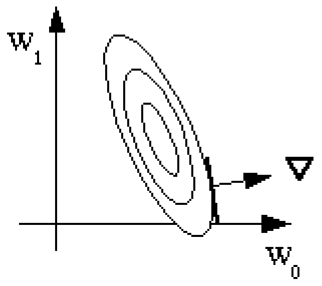
$$E\left[\varepsilon\ \right] \quad r \qquad\qquad {}^{T} \qquad\qquad {}^{T}R$$

If R is positive definite, the error surface $E\left[\varepsilon\ \right]\ w\ \ w \qquad w_M$ is a unimodal "bowl" in $\ ^{N}$.



The problem is to find the bottom of the bowl. In an adaptive filter context, the shape and bottom of the bowl may drift slowly with time; hopefully slow enough that the adaptive algorithm can track it.

For a quadratic error surface, the bottom of the bowl can be found in one step by computing R . Most modern nonlinear optimization methods (which are used, for example, to solve the L^P optimal IIR filter design problem!) locally approximate a nonlinear function with a second-order (quadratic) Taylor series approximation and step to the bottom of this quadratic approximation on each iteration. However, an older and simpler appraoch to nonlinear optimaztion exists, based on **gradient descent**.
Contour plot of ε-squared



The idea is to iteratively find the minimizer by computing the gradient of the error function: E $\frac{E[\varepsilon\ ]}{w_i}$. The gradient is a vector in ^{M} pointing in the steepest uphill direction on the error surface at a given point ^{i}, with having a magnitude proportional to the slope of the error surface in this steepest direction.

By updating the coefficient vector by taking a step **opposite** the gradient direction : ^{i} ^{i} μ ^{i}, we go (locally) "downhill" in the steepest direction, which seems to be a sensible way to iteratively solve a nonlinear optimization problem. The performance obviously depends on μ; if μ is too large, the iterations could bounce back and forth up out of the bowl. However, if μ is too small, it could take many iterations to approach the bottom. We will determine criteria for choosing μ later.

In summary, the gradient descent algorithm for solving the Weiner filter problem is:

$$W$$

$$i$$

$$^i \quad P \quad RW^i$$

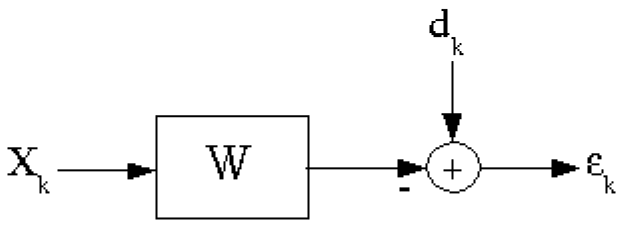
$$W^i \quad W^i \quad \mu \quad ^i$$

$$W \quad W$$

The gradient descent idea is used in the LMS adaptive fitler algorithm. As presented, this alogrithm costs $O\left(M\ \right)$ computations per iteration and doesn't appear very attractive, but LMS only requires $O\ M$ computations and is stable, so it is very attractive when computation is an issue, even thought it converges more slowly then the RLS algorithms we have discussed so far.

The LMS Adaptive Filter Algorithm

Recall the Weiner filter problem



$\{x_k\}$, $\{d_k\}$ jointly wide sense stationary

Find W minimizing $E\left[{\varepsilon_k}^2\right]$

$$\varepsilon_k = d_k - y_k = d_k - \sum_{i=0}^{M-1} w_i x_{k-i} = d_k - {X^k}^T W^k$$

$$X^k = \begin{matrix} x_k \\ x_{k-1} \\ \vdots \\ x_{k-M+1} \end{matrix}$$

$$W^k = \begin{matrix} w_0^k \\ w_1^k \\ \vdots \\ w_{M-1}^k \end{matrix}$$

The superscript denotes absolute time, and the subscript denotes time or a vector index.

the solution can be found by setting the gradient 0

Equation:

- -

$$\begin{aligned}\nabla^k &= \frac{\partial E\left[{\varepsilon_k}^2\right]}{\partial W} \\ &= E\left[2\varepsilon_k\left(-X^k\right)\right] \\ &= E\left[-2\left(d_k - {X^k}^T W_k\right) X^k\right] \\ &= -\left(2E\left[d_k X^k\right]\right) + E\left[{X^k}^T\right] W \\ &= 2P + 2RW\end{aligned}$$

$$\Rightarrow \left(W_{\text{opt}} = R^{-1} P\right)$$

Alternatively, W_{opt} can be found iteratively using a gradient descent technique

$$W^{k+1} = W^k - \mu \nabla^k$$

In practice, we don't know R and P exactly, and in an adaptive context they may be slowly varying with time.

To find the (approximate) Wiener filter, some approximations are necessary. As always, the key is to make the **right** approximations!

Note: Approximate R and P: ⇒ RLS methods, as discussed last time.

Note: Approximate the gradient!

$$\nabla^k = \frac{\partial E\left[{\varepsilon_k}^2\right]}{\partial W}$$

Note that ${\varepsilon_{\mathrm{k}}}^2$ itself is a very noisy approximation to $E\left[{\varepsilon_{\mathrm{k}}}^2\right]$. We can get a noisy approximation to the gradient by finding the gradient of ${\varepsilon_{\mathrm{k}}}^2$! Widrow and Hoff first published the LMS algorithm, based on this clever idea, in 1960.

$$\widehat{\nabla^k} = \frac{\partial {\varepsilon_k}^2}{\partial W} = 2\varepsilon_k \frac{\partial \left(d_k - {W^k}^T X^k\right)}{\partial W} = 2\varepsilon_k \left(-X^k\right) = -\left(2\varepsilon_k X^k\right)$$

This yields the LMS adaptive filter algorithm

Example:
The LMS Adaptive Filter Algorithm

1. $y_k = {W^k}^T X^k = \sum_{i=0}^{M-1} w_i^k x_{k-i}$
2. $\varepsilon_k = d_k - y_k$
3. $W^{k+1} = W^k - \mu\widehat{\nabla^k} = W^k - \mu\left(-2\varepsilon_k X^k\right) = W^k + 2\mu\varepsilon_k X^k$ ($w_i^{k+1} = w_i^k + 2\mu\varepsilon_k x_{k-i}$)

The LMS algorithm is often called a **stochastic gradient** algorithm, since $\widehat{\nabla^k}$ is a noisy gradient. This is **by far** the most commonly used adaptive filtering algorithm, because

1. it was the first
2. it is very simple
3. in practice it works well (except that sometimes it converges slowly)
4. it requires relatively litle computation
5. it updates the tap weights every sample, so it continually adapts the filter
6. it tracks slow changes in the signal statistics well

Computational Cost of LMS

To Compute ⇒	y_k	ε_k	W^{k+1}	**= Total**
multiplies	M	0	$M+1$	$2M+1$
adds	$M-1$	1	M	$2M$

So the LMS algorithm is $O(M)$ per sample. In fact, it is nicely balanced in that the filter computation and the adaptation require the same amount of computation.

Note that the parameter μ plays a very important role in the LMS algorithm. It can also be varied with time, but usually a constant μ ("convergence weight facor") is used, chosen after experimentation for a given application.

Tradeoffs

large μ: fast convergence, fast adaptivity

small μ: accurate W → less misadjustment error, stability

First Order Convergence Analysis of the LMS Algorithm

Analysis of the LMS algorithm

It is important to analyze the LMS algorithm to determine under what conditions it is stable, whether or not it converges to the Wiener solution, to determine how quickly it converges, how much degredation is suffered due to the noisy gradient, etc. In particular, we need to know how to choose the parameter μ.

Mean of W

does W^k, $k \to \infty$ approach the Wiener solution? (since W^k is always somewhat random in the approximate gradient-based LMS algorithm, we ask whether the expected value of the filter coefficients converge to the Wiener solution)
Equation:

$$\begin{aligned} E\left[W^{k+1}\right] &= W^{k+1} \\ &= E\left[W^k + 2\mu\varepsilon_k X^k\right] \\ &= W^k + 2\mu E\left[d_k X^k\right] + 2\mu E\left[-\left(\left({W^k}^T X^k\right) X^k\right)\right] \\ &= W^k + 2\mu P + -\left(2\mu E\left[\left({W^k}^T X^k\right) X^k\right]\right) \end{aligned}$$

Patently False Assumption

X^k and X^{k-i}, X^k and d^{k-i}, and d_k and d_{k-i} are statistically independent, $i \neq 0$. This assumption is obviously false, since X^{k-1} is the same as X^k except for shifting down the vector elements one place and adding one new sample. We make this assumption because otherwise it becomes extremely difficult to analyze the LMS algorithm. (First good analysis not making this assumption: Macchi and Eweda) Many simulations and much practical experience has shown that the results one obtains with analyses based on the patently false assumption above are quite accurate in most situations

With the independence assumption, W^k (which depends only on previous X^{k-i}, d^{k-i}) is statitically independent of X^k, and we can simplify $E\left[\left({W^k}^T X^k\right) X^k\right]$

Now $\left({W^k}^T X^k\right) X^k$ is a vector, and
Equation:

$$\begin{aligned} E\left[\left({W^k}^T X^k\right) X^k\right] &= E \begin{matrix} \vdots \\ \sum_{i=0}^{M-1} w_i^k x_{k-i} x_{k-j} \\ \vdots \end{matrix} \\ &= \begin{matrix} \vdots \\ \sum_{i=0}^{M-1} E\left[w_i^k x_{k-i} x_{k-j}\right] \\ \vdots \end{matrix} \\ &= \begin{matrix} \vdots \\ \sum_{i=0}^{M-1} \left(w_i^k\right) E\left[x_{k-i} x_{k-j}\right] \\ \vdots \end{matrix} \\ &= \begin{matrix} \vdots \\ \sum_{i=0}^{M-1} w_i^k r_{\mathrm{xx}}(i-j) \\ \vdots \end{matrix} \\ &= R W^k \end{aligned}$$

where $R = E\left[X^k {X^k}^T\right]$ is the data correlation matrix.

Putting this back into our equation
Equation:

$$\begin{aligned} W^{k+1} &= W^k + 2\mu P + -\left(2\mu R W^k\right) \\ &= I W^k + 2\mu P \end{aligned}$$

Now **if** $W^{k \to \infty}$ converges to a vector of finite magnitude ("convergence in the mean"), what does it converge to?

If W^k converges, then as $k \to \infty$, $W^{k+1} \simeq W^k$, and

$$W^\infty = IW^\infty + 2\mu P$$

$$2\mu RW^\infty = 2\mu P$$

$$RW^\infty = P$$

or

$$W_{\text{opt}} = R^{-1}P$$

the Wiener solution!

So the LMS algorithm, **if** it converges, gives filter coefficients which on average are the Wiener coefficients! This is, of course, a desirable result.

First-order stability

But does W^k converge, or under what conditions?

Let's rewrite the analysis in term of V^k, the "mean coefficient error vector"

$V^k = W^k - W_{\text{opt}}$, where W_{opt} is the Wiener filter

$$W^{k+1} = W^k - 2\mu RW^k + 2\mu P$$

$$W^{k+1} - W_{\text{opt}} = W^k - W_{\text{opt}} + -\left(2\mu RW^k\right) + 2\mu RW_{\text{opt}} - 2\mu RW_{\text{opt}} + 2\mu P$$

$$V^{k+1} = V^k - 2\mu R V^k + -\left(2\mu R W_{\text{opt}}\right) + 2\mu P$$

Now $W_{\text{opt}} = R^{-1}$, so

$$V^{k+1} = V^k - 2\mu R V^k + -\left(2\mu R R^{-1} P\right) + 2\mu P = (I - 2\mu R) V^k$$

We wish to know under what conditions $V^{k \to \infty} \to \bar{0}$?

Linear Algebra Fact

Since R is positive definite, real, and symmetric, all the eigenvalues are real and positive. Also, we can write R as $Q^{-1} \Lambda Q$, where $\mathit{\Lambda}$ is a diagonal matrix with diagonal entries λ_i equal to the eigenvalues of R, and Q is a unitary matrix with rows equal to the eigenvectors corresponding to the eigenvalues of R.

Using this fact,

$$V^{k+1} = \left(I - 2\mu\left(Q^{-1} \Lambda Q\right)\right) V^k$$

multiplying both sides through on the left by Q: we get

$$Q V^{k+1} = (Q - 2\mu \mathit{\Lambda} Q) V^k = (1 - 2\mu \mathit{\Lambda}) Q V^k$$

Let $V' = QV$:

$$V'^{k+1} = (1 - 2\mu \mathit{\Lambda}) V'^{k}$$

Note that V' is simply V in a rotated coordinate set in $\mathbb{R}^m$, so convergence of V' implies convergence of V.

Since $1 - 2\mu \mathit{\Lambda}$ is diagonal, all elements of V' evolve independently of each other. Convergence (stability) bolis down to whether all M of these scalar, first-order difference equations are stable, and thus $\to (0)$.

$$\forall i, i = [1, 2, \ldots, M] : \left(V_i'^{k+1} = (1 - 2\mu \lambda_i) V_i'^{k}\right)$$

These equations converge to zero if $|1-2\mu\lambda_i|<1$, or $\forall i:(|\mu\lambda_i|<1)$ μ and λ_i are positive, so we require $\forall i:\left(\mu<\frac{1}{\lambda_i}\right)$ so for convergence in the mean of the LMS adaptive filter, we require
Equation:

$$\mu<\frac{1}{\lambda_{\max}}$$

This is an elegant theoretical result, but in practice, we may not know $\lambda_{\max}$, it may be time-varying, and we certainly won't want to compute it. However, another useful mathematical fact comes to the rescue...

$$\operatorname{tr}(R)=\sum_{i=1}^{M} r_{\text{ii}}=\sum_{i=1}^{M}\lambda_i\geq\lambda_{\max}$$

Since the eigenvalues are all positive and real.

For a correlation matrix, $\forall i, i\in\{1,M\}:(r_{\text{ii}}=r(0))$. So $\operatorname{tr}(R)=Mr(0)=ME[x_kx_k]$. We can easily estimate $r(0)$ with $O(1)$ computations/sample, so in practice we might require

$$\mu<\frac{1}{Mr(0)}$$

as a conservative bound, and perhaps adapt μ accordingly with time.

Rate of convergence

Each of the modes decays as

$$(1-2\mu\lambda_i)^k$$

Note: The **initial** rate of convergence is dominated by the fastest mode $1-2\mu\lambda_{\max}$. This is not surprising, since a dradient descent method goes "downhill" in the steepest direction

Note: The **final** rate of convergence is dominated by the slowest mode $1 - 2\mu\lambda_{\min}$. For small $\lambda_{\min}$, it can take a long time for LMS to converge.

Note that the convergence behavior depends on the data (via R). LMS converges relatively quickly for roughly equal eigenvalues. Unequal eigenvalues slow LMS down a lot.

Second-order Convergence Analysis of the LMS Algorithm and Misadjustment Error

Convergence of the mean (first-order analysis) is insufficient to guarantee desirable behavior of the LMS algorithm; the variance could still be infinite. It is important to show that the variance of the filter coefficients is finite, and to determine how close the average squared error is to the minimum possible error using an exact Wiener filter.

Equation:

$$\begin{aligned} E\left[{\varepsilon_k}^2\right] &= E\left[\left(d_k - {W^k}^T X^k\right)^2\right] \\ &= E\left[{d_k}^2 - 2 d_k {X^k}^T W^k - {W^k}^T X^k {X^k}^T W^k\right] \\ &= r_{\mathrm{dd}}(0) - 2{W^k}^T P + {W^k}^T R W^k \end{aligned}$$

The minimum error is obtained using the Wiener filter

$$W_{\mathrm{opt}} = R^{-1} P$$

Equation:

$$\begin{aligned} {\varepsilon_{\min}}^2 &= E\left[\varepsilon^2\right] \\ &= \left(r_{\mathrm{dd}}(0) - 2P^T R^{-1} P + P^T R^{-1} R R^{-1} P\right) \\ &= r_{\mathrm{dd}}(0) - P^T R^{-1} P \end{aligned}$$

To analyze the average error in LMS, write [link] in terms of $V' = Q[W - W_{\mathrm{opt}}]$, where $Q^{-1} \Lambda Q = R$

Equation:

$$\begin{aligned} E\left[{\varepsilon_k}^2\right] &= r_{\mathrm{dd}}(0) - 2{W^k}^T P + {W^k}^T R W^k + \left(-\left({W^k}^T R W_{\mathrm{opt}}\right)\right) - {W_{\mathrm{opt}}}^T R W^k + {W_{\mathrm{opt}}}^T R W_{\mathrm{opt}} + {W^k}^T R \\ &= r_{\mathrm{dd}}(0) + {V^k}^T R V^k - P^T R^{-1} P \\ &= {\varepsilon_{\min}}^2 + {V^k}^T R V^k \\ &= {\varepsilon_{\min}}^2 + {V^k}^T Q^{-1} Q R Q^{-1} Q V^k \\ &= {\varepsilon_{\min}}^2 + {V^{\text{'k}}}^T \Lambda V^{\text{'k}} \end{aligned}$$

$$E\left[{\varepsilon_k}^2\right] = {\varepsilon_{\min}}^2 + \sum_{j=0}^{N-1} \lambda_j E\left[{v_j^{\text{'k}}}^2\right]$$

So we need to know $E\left[{v_j^{\text{'k}}}^2\right]$, which are the diagonal elements of the covariance matrix of $V^{\text{'k}}$, or $E\left[V^{\text{'k}} {V^{\text{'k}}}^T\right]$.

From the LMS update equation

$$W^{k+1} = W^k + 2\mu \varepsilon_k X^k$$

we get

$$V^{\text{'k+1}} = W^{\text{'k}} + 2\mu \varepsilon_k Q X^k$$

Equation:

$$\begin{aligned}\mathcal{V}^{k+1} &= E\left[V'^{k+1}V'^{k+1T}\right] \\ &= E\left[4\mu^2{\varepsilon_k}^2QX^kX^{kT}Q^T\right] \\ &= \mathcal{V}^k + 2\mu\left(\varepsilon_kQX^kV'^{\text{k}T}\right) + 2\mu\left(\varepsilon_kV'^{\text{k}}X^{kT}Q^T\right) + 4\mu^2E\left[{\varepsilon_k}^2QX^kX^{kT}Q^T\right]\end{aligned}$$

Note that

$$\varepsilon_k = d_k - W^{kT}X^k = d_k - {W_{\text{opt}}}^T - V'^{\text{k}T}QX^k$$

so
Equation:

$$\begin{aligned}E\left[\varepsilon_kQX^kV'^{\text{k}T}\right] &= E\left[d_kQX^kV'^{\text{k}T} - {W_{\text{opt}}}^TX^kQX^kV'^{\text{k}T} - V'^{\text{k}T}QX^kV'^{\text{k}T}\right] \\ &= 0 + 0 - \left(QX^kX^{kT}Q^TV'^{\text{k}}V'^{\text{k}T}\right) \\ &= -\left(QE\left[X^kX^{kT}\right]Q^TE\left[V'^{\text{k}}V'^{\text{k}T}\right]\right) \\ &= -\left(\Lambda\mathcal{V}^k\right)\end{aligned}$$

Note that the Patently False independence Assumption was invoked here.

To analyze $E\left[{\varepsilon_k}^2QX^kX^{kT}Q^T\right]$, we make yet another obviously false assumptioon that ${\varepsilon_k}^2$ and X^k are statistically independent. This is obviously false, since $\varepsilon_k = d_k - W^{kT}X^k$. Otherwise, we get 4th-order terms in X in the product. These can be dealt with, at the expense of a more complicated analysis, if a particular type of distribution (such as Gaussian) is assumed. See, for example Gardner. A questionable justification for this assumption is that as $W^k \simeq W_{\text{opt}}$, W^k becomes uncorrelated with X^k (if we invoke the original independence assumption), which tends to randomize the error signal relative to X^k. With this assumption,

$$E\left[{\varepsilon_k}^2QX^kX^{kT}Q^T\right] = E\left[{\varepsilon_k}^2\right]E\left[QX^kX^{kT}Q^T\right] = E\left[{\varepsilon_k}^2\right]\Lambda$$

Now

$${\varepsilon_k}^2 = {\varepsilon_{\text{min}}}^2 + V'^{\text{k}T}\Lambda V'^{\text{k}}$$

so
Equation:

$$\begin{aligned}E\left[{\varepsilon_k}^2\right] &= {\varepsilon_{\text{min}}}^2 + E\left[\sum_j \lambda_j {V_j'^{\text{k}}}^2\right] \\ &= {\varepsilon_{\text{min}}}^2 + \sum_j \lambda_j \mathcal{V}_{\text{jj}}^k\end{aligned}$$

Thus, [link] becomes
Equation:

$$\mathcal{V}^{k+1} = I\mathcal{V}^k + 4\mu^2\sum_j \lambda_j\mathcal{V}_{\text{jj}}^k\Lambda + 4\mu^2{\varepsilon_{\text{min}}}^2\Lambda$$

Now **if** this system is stable and converges, it converges to $\mathcal{V}^\infty = \mathcal{V}^{\infty+1}$

$$\Rightarrow \left(4\mu\Lambda\mathscr{V}^{\infty} = 4\mu^2\left(\sum_j \lambda_j \mathscr{V}_{\mathrm{jj}} + {\varepsilon_{\min}}^2\right)\Lambda\right)$$

$$\Rightarrow \left(\mathscr{V}^{\infty} = \mu\left(\sum_j \lambda_j \mathscr{V}_{\mathrm{jj}} + {\varepsilon_{\min}}^2\right)I\right)$$

So it is a diagonal matrix with all elements on the diagonal equal:

Then

$$\mathscr{V}_{\mathrm{ii}}^{\infty} = \mu\left(\mathscr{V}_{\mathrm{ii}}^{\infty}\sum_j \lambda_j + {\varepsilon_{\min}}^2\right)$$

$$\mathscr{V}_{\mathrm{ii}}^{\infty}\left(1 - \mu\sum_j \lambda_j\right) = \mu{\varepsilon_{\min}}^2$$

$$\mathscr{V}_{\mathrm{ii}}^{\infty} = \frac{\mu{\varepsilon_{\min}}^2}{1 - \mu\sum_j \lambda_j}$$

Thus the error in the LMS adaptive filter after convergence is
Equation:

$$\begin{aligned} E\left[{\varepsilon_{\infty}}^2\right] &= {\varepsilon_{\min}}^2 + E\left[V'^{\infty}\lambda V'^{\infty}\right] \\ &= {\varepsilon_{\min}}^2 + \frac{\mu{\varepsilon_{\min}}^2\sum_j \lambda_j}{1-\mu\sum_j \lambda_j} \\ &= {\varepsilon_{\min}}^2\frac{1}{1-\mu\sum_j \lambda_j} \\ &= {\varepsilon_{\min}}^2\frac{1}{1-\mu\,\mathrm{tr}\,(R)} \\ &= {\varepsilon_{\min}}^2\frac{1}{1-\mu r_{\mathrm{xx}}(0)\mathrm{N}} \end{aligned}$$

Equation:

$$E\left[{\varepsilon_{\infty}}^2\right] = {\varepsilon_{\min}}^2\frac{1}{1-\mu\mathrm{N}{\sigma_x}^2}$$

$1 - \mu\mathrm{N}{\sigma_x}^2$ is called the **misadjustment factor**. Oftern, one chooses μ to select a desired misadjustment factor, such as an error 10% higher than the Wiener filter error.

2nd-Order Convergence (Stability)

To determine the range for μ for which [link] converges, we must determine the μ for which the matrix difference equation converges.

$$\mathscr{V}^{k+1} = I\mathscr{V}^{k} + 4\mu^2\sum_j \lambda_j \mathscr{V}_{\mathrm{jj}}^{k}\Lambda + 4\mu^2{\varepsilon_{\min}}^2\Lambda$$

The off-diagonal elements each evolve independently according to $\mathscr{V}_{\mathrm{ij}}^{k+1} = 1 - 4\mu\lambda_i\mathscr{V}_{\mathrm{ij}}^{k}$ These terms will decay to zero if $\forall i : (4\mu\lambda_i < 2)$, or $\mu < \frac{1}{2\lambda_{\max}}$

The diagonal terms evolve according to

$$\mathscr{V}_{\mathrm{ii}}^{k+1} = 1\mathscr{V}_{\mathrm{ii}}^{k} + 4\mu^2\lambda_i \sum_j \lambda_j \mathscr{V}_{\mathrm{jj}}^{k} + 4\mu^2 {\varepsilon_{\min}}^2 \lambda_i$$

For the homoegeneous equation

$$\mathscr{V}_{\mathrm{ii}}^{k+1} = 1\mathscr{V}_{\mathrm{ii}}^{k} + 4\mu^2\lambda_i \sum_j \lambda_j \mathscr{V}_{\mathrm{jj}}^{k}$$

for $1 - 4\mu\lambda_i$ positive,
Equation:

$$\mathscr{V}_{\mathrm{ii}}^{k+1} \leq 1\mathscr{V}_{\mathrm{iimax}}^{k} + 4\mu^2\lambda_i \sum_j \lambda_j \mathscr{V}_{\mathrm{jjmax}}^{k} = \left(1 - 4\mu\lambda_i + 4\mu^2\lambda_i \sum_j \lambda_j\right)\mathscr{V}_{\mathrm{jjmax}}^{k}$$

$\mathscr{V}_{\mathrm{ii}}^{k+1}$ will be strictly less than $\mathscr{V}_{\mathrm{jjmax}}^{k}$ for

$$1 - 4\mu\lambda_i + 4\mu^2\lambda_i \sum_j \lambda_j < 1$$

or

$$4\mu^2\lambda_i \sum_j \lambda_j < 4\mu\lambda_i$$

or
Equation:

$$\begin{aligned} \mu < \frac{1}{\sum_j \lambda_j} &= \frac{1}{\mathrm{tr}\,(R)} \\ &= \frac{1}{N r_{\mathrm{xx}}(0)} \\ &= \frac{1}{N{\sigma_x}^2} \end{aligned}$$

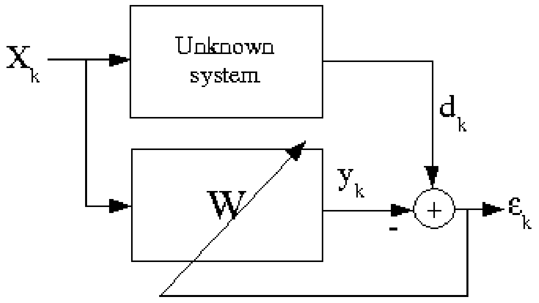
This is a more rigorous bound than the first-order bounds. Ofter engineers choose μ a few times smaller than this, since more rigorous analyses yield a slightly smaller bound. $\mu = \frac{\mu}{3N{\sigma_x}^2}$ is derived in some analyses assuming Gaussian x_k, d_k.

Applications of Adaptive Filters

Adaptive filters can be configured in a surprisingly large variety of ways for application in a large number of applications. The same LMS algorithm can be used to adapt the filter coefficients in most of these configurations; only the "plumbing" changes.

Adaptive System Identification

Note: To approximate an unknown system (or the behavior of that system) as closely as possible



The optimal solution is $R^{-1}P = W$

Suppose the unknown system is a causal, linear time-invariant filter:

$$d_k = x_k * h_k = \sum_{i\ \ 0}^{\infty} x_{k-i} h_i$$

Now
Equation:

$$\begin{aligned}
P &= (E[d_k x_{k-j}]) \\
&= (E[\ {}^{\infty}_{i\ \ 0}\, x_{k-i} h_i x_{k-j}]) \\
&= (\ {}^{\infty}_{i\ \ 0}\, h_i E[x_{k-i} x_{k-j}]) \\
&= (\ {}^{\infty}_{i\ \ 0}\, r_{\mathrm{xx}}(j-i)) \\
&= \begin{array}{ccccccccc}
r_{\mathrm{xx}}(0) & r(1) & \dots & \dots & r(M-1) & | & r(M) & r(M+1) & \dots \\
r(1) & r(0) & \ddots & \ddots & \vdots & | & \vdots & \vdots & \dots \\
r(2) & r(1) & \ddots & \ddots & \vdots & | & \vdots & \vdots & \dots \\
\vdots & \vdots & \dots & r(0) & r(1) & | & r(2) & r(3) & \dots \\
r(M-1) & r(M-2) & \dots & r(1) & r(0) & | & r(1) & r(2) & \dots
\end{array}
\begin{array}{c} h(0) \\ h(1) \\ h(2) \\ \vdots \end{array}
\end{aligned}$$

If the adaptive filter H is a length-M FIR filter ($h(m) = h(m+1) = \dots = 0$), this reduces to

$$P = Rh^{-1}$$

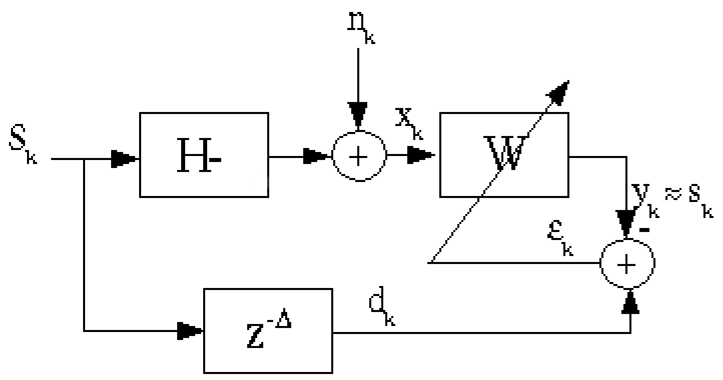
and

$$W_{\text{opt}} = R^{-1}P = R^{-1}\left(R\quad\right) =$$

FIR adaptive system identification thus converges in the mean to the corresponding M samples of the impulse response of the unknown system.

Adaptive Equalization

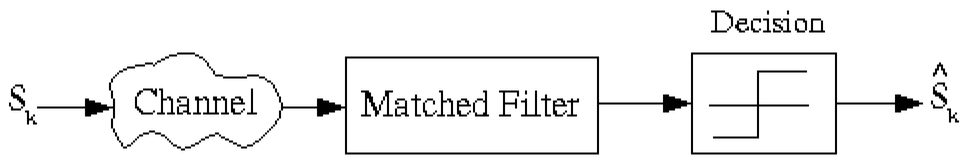
Note: Design an approximate inverse filter to cancel out as much distortion as possible.



In principle, , or —, so that the overall response of the top path is approximately . However, limitations on the form of (FIR) and the presence of noise cause the equalization to be imperfect.

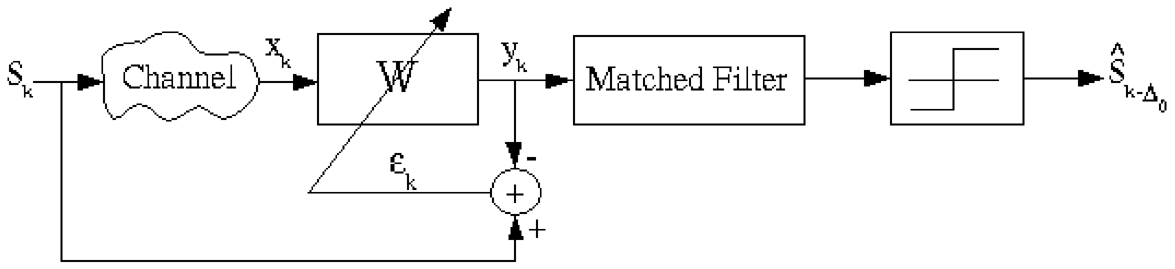
Important Application

Channel equalization in a digital communication system.



If the channel distorts the pulse shape, the matched filter will no longer be matched, intersymbol interference may increase, and the system performance will degrade.

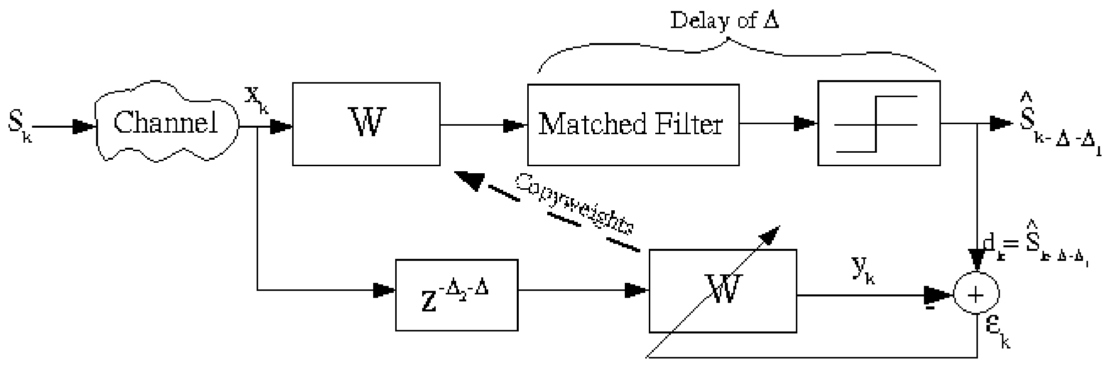
An adaptive filter is often inserted in front of the matched filter to compensate for the channel.



This is, of course, unrealizable, since we do not have access to the original transmitted signal, .

There are two common solutions to this problem:

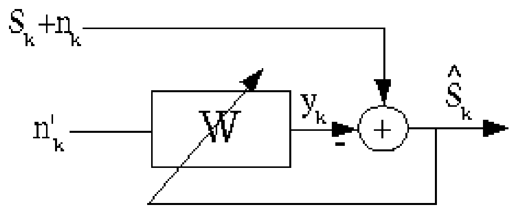
1. Periodically broadcast a known **training signal**. The adaptation is switched on only when the training signal is being broadcast and thus is known.
2. Decision-directed feedback: If the overall system is working well, then the output should almost always equal . We can thus use our received digital communication signal as the desired signal, since it has been cleaned of noise (we hope) by the nonlinear threshold device!
   Decision-directed equalizer



As long as the error rate in   is not too high (say   ), this method works. Otherwise,   is so inaccurate that the adaptive filter can never find the Wiener solution. This method is widely used in the telephone system and other digital communication networks.

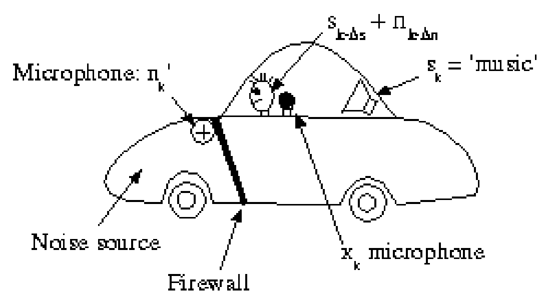
Adaptive Interference (Noise) Cancellation

Note: Automatically eliminate unwanted interference in a signal.

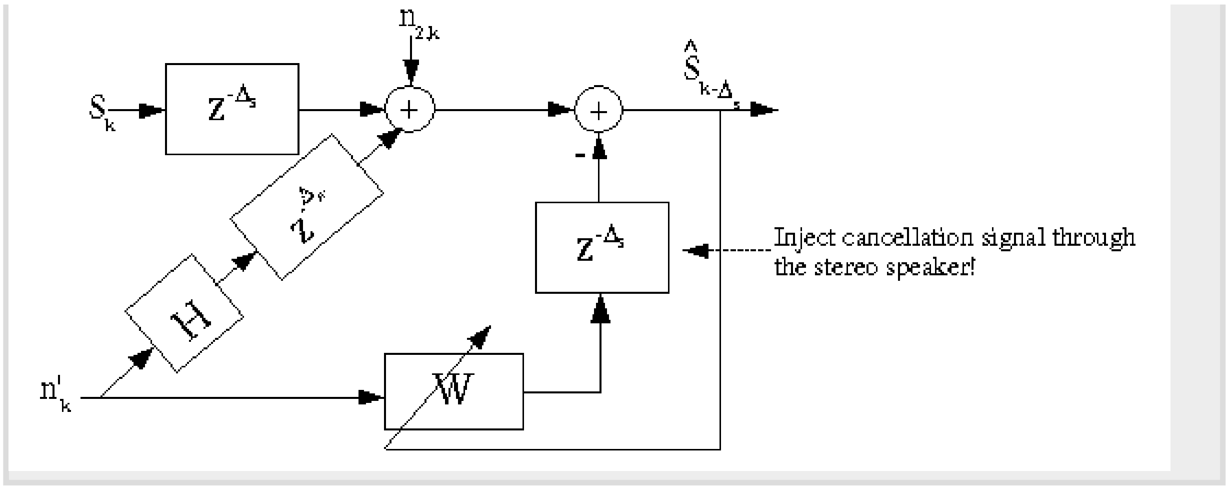


The object is to subtract out as much of the noise as possible.

Example:
Engine noise cancellation in automobiles



The firewall attenuates and filters the noise reaching the listener's ear, so it is not the same as . There is also a delay due to acoustic propagation in the air. For maximal cancellation, an adaptive filter is thus needed to make as similar as possible to the delayed .

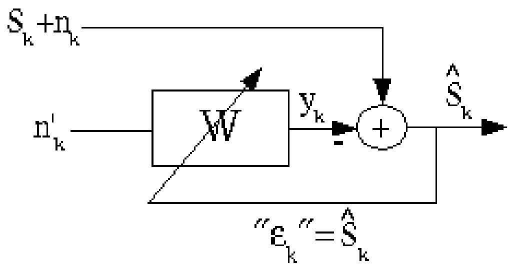


Exercise:

Problem:

What conditions must we impose upon the microphone locations for this to work? (Think causality and physics!)

Analysis of the interference cancellor

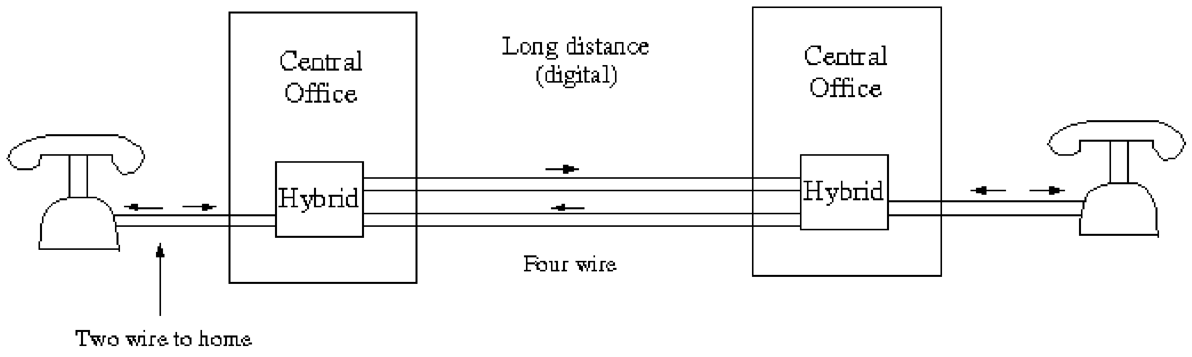


We assume , , and are zero-mean signals, and that is independent of and . Then

Since the input signal has no information about in it, minimizing can only affect the second term, which is the standard Wiener filtering problem, with solution

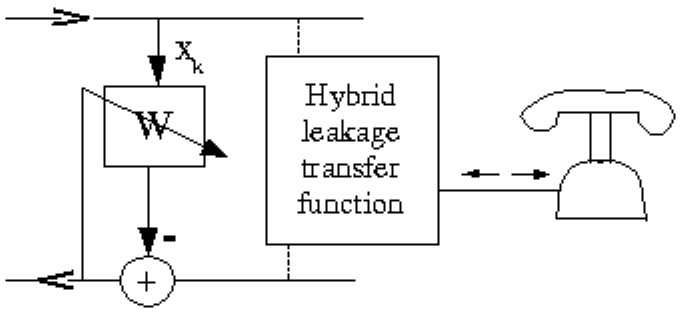
Adaptive Echo Cancellation

An adaptive echo canceller is a specific type of adaptive interference canceller that removes echos. Adaptive echo cancellers are found in all modern telephone systems.



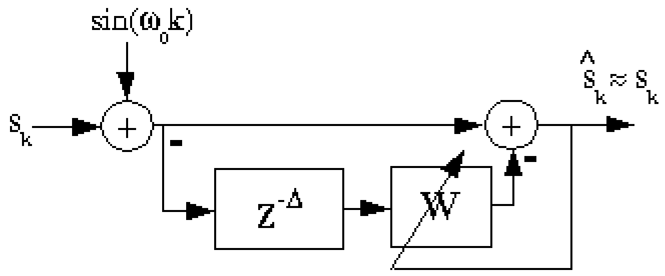
The hybrid is supposed to split the opposite-going waves, but typically achieves only about 15dB of suppression. This signal will eventually reach the other end and be coupled back, with a long delay, to the original source, which gives a very annoying echo.

Echo canceller



Because the input to the adaptive echo canceller contains only the signal from the far end that will echo off of the hybrid, it cancels the echo while passing the near-end signal as desired.

Narrowband interference canceller



A sinusoid is predictable samples ahead, whereas may not be, so the sinusoid can be cancelled using the adaptive system in the Figure. This is another special case of the adaptive interference canceller in which the noise reference input is a delayed version of the primary (signal plus noise) input. Note that must be large enough so that and are uncorrelated, or some of the signal will be cancelled as well!

Exercise:

Problem:

How would you construct an "adaptive line enhancer" that preserves the sinusoids but cancels the uncorrelated noise?

Other Applications

- Adaptive array processing
- Adaptive control
- etc...

Beyond LMS: an overview of other adaptive filter algorithms

RLS algorithms

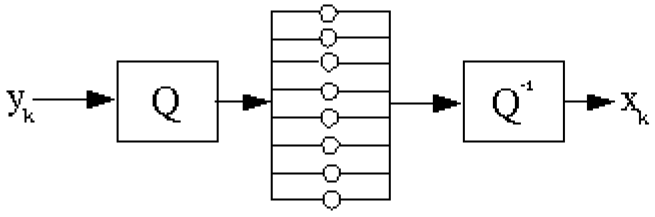
FIR adaptive filter algorithms with faster convergence. Since the Wiener solution can be obtained on one step by computing $W \qquad R \quad P$, most RLS algorithms attept to estimate R and P and compute W from these.

There are a number of $O \; N$ algorithms which are stable and converge quickly. A number of $O \; N$ algorithms have been proposed, but these are all unstable except for the lattice filter method. This is described to some extent in the text. The adaptive lattice filter converges quickly and is stable, but reportedly has a very high noise floor.

Many of these approaches can be thought of as attempting to "orthogonalize" R, or to rotate the data or filter coefficients to a domain where R is diagonal, then doing LMS in each dimension **separately**, so that a fast-converging step size can be chosen in all directions.

Frequency-domain methods

Frequency-domain methods implicitly attempt to do this:



If QRQ is a diagonal matrix, this yields a fast algorithm. If Q is chosen as an FFT matrix, each channel becomes a different frequency bin. Since R is Toeplitz and not a circulant, the FFT matrix will not exactly diagonalize R, but in many cases it comes very close and frequency domain methods converge very quickly. However, for some R they perform no better than LMS. By using an FFT, the transformation Q becomes inexpensive $O\ N \qquad N$. If one only updates on a block-by-block basis (once per N samples), the frequency domain methods only cost $O \qquad N$ computations per sample. which can be important for some applications with large N. (Say 16,000,000)

Adaptive IIR filters

Adaptive IIR filters are attractive for the same reasons that IIR filters are attractive: many fewer coefficients may be needed to achieve the desired performance in some applications. However, it is more difficult to develop stable IIR algorithms, they can converge **very** slowly, and they are susceptible to local minima. Nonetheless, adaptive IIR algorithms are used in some applications (such as low frequency noise cancellation) in which the need for IIR-type responses is great. In some cases, the exact algorithm used by a company is a tightly guarded trade secret.

Most adaptive IIR algorithms minimize the **prediction** error, to linearize the estimation problem, as in deterministic or block linear prediction.

$$y_k = \sum_{n=1}^{L} v_n^k y_{k-n} + \sum_{n=0}^{L} w_n^k x_{k-n}$$

Thus the coefficient vector is

$$W_k = \begin{matrix} v_1^k \\ v_2^k \\ \vdots \\ v_L^k \\ w_0^k \\ w_1^k \\ \vdots \\ w_L^k \end{matrix}$$

and the "signal" vector is

$$U_k = \begin{matrix} y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-L} \\ x_k \\ x_{k-1} \\ \vdots \\ x_{k-L} \end{matrix}$$

The error is

$$\varepsilon_k = d_k - y_k = d_k - {W_k}^T U_k$$

An LMS algorithm can be derived using the approximation $E\left[{\varepsilon_k}^2\right] = {\varepsilon_k}^2$ or

$$\widehat{\nabla}_k = \frac{\partial {\varepsilon_k}^2}{\partial W_k} = 2\varepsilon_k \frac{\partial \varepsilon_k}{\partial W_k} = 2\varepsilon_k \begin{matrix} \frac{\partial \varepsilon_k}{\partial v_1^k} \\ \vdots \\ \frac{\partial w_1^k}{\partial \varepsilon_k} \\ \vdots \end{matrix} = -2\varepsilon_k \begin{matrix} \frac{\partial y_k}{\partial v_1^k} \\ \vdots \\ \frac{\partial y_k}{\partial v_L^k} \\ \frac{\partial y_k}{\partial w_0^k} \\ \vdots \\ \frac{\partial y_k}{\partial w_L^k} \end{matrix}$$

Now

$$\frac{\partial y_k}{\partial v_i^k} = \frac{\partial \left(\sum_{n=1}^{L} v_n^k y_{k-n} + \sum_{n=0}^{L} w_n^k x_{k-n}\right)}{\partial v_i^k} = y_{k-n} + \sum_{n=1}^{L} v_n^k \frac{\partial y_{k-n}}{\partial v_i^k} + 0$$

$$\frac{\partial y_k}{\partial w_i^k} = \frac{\partial \left(\sum_{n=1}^{L} v_n^k y_{k-n} + \sum_{n=0}^{L} w_n^k x_{k-n}\right)}{\partial w_i^k} = \sum_{n=1}^{L} v_n^k \frac{\partial y_{k-n}}{\partial w_i^k} + x_{k-n}$$

Note that these are difference equations in $\frac{\partial y_k}{\partial v_i^k}$, $\frac{\partial y_k}{\partial w_i^k}$: call them $\alpha_i^k = \frac{\partial y_k}{\partial w_i^k}$, $\beta_i^k = \frac{\partial y_k}{\partial v_i^k}$, then $\widehat{\nabla}_k = \begin{pmatrix} \beta_1^k & \beta_2^k & \dots & \beta_L^k & \alpha_0^k & \dots & \alpha_L^k \end{pmatrix}^T$, and the IIR LMS algorithm becomes

$$y_k = {W_k}^T U_k$$

$$\alpha_i^k = x_{k-i} + \sum_{j=1}^{L} v_j^k \alpha_i^{k-j}$$

$$\beta_i^k = y_{k-i} + \sum_{j=1}^{L} v_j^k \beta_i^{k-j}$$

$$\widehat{\nabla}_k = -2\varepsilon_k \begin{pmatrix} \beta_1^k & \beta_2^k & \dots & \alpha_0^k & \alpha_1^k & \dots & \alpha_L^k \end{pmatrix}^T$$

and finally

$$W_{k+1} = W_k - U\widehat{\nabla}_k$$

where the μ may be **different** for the different IIR coefficients. Stability and convergence rate depends on these choices, of course. There are a number of variations on this algorithm.

Due to the slow convergence and the difficulties in tweaking the algorithm parameters to ensure stability, IIR algorithms are used only if there is an overriding need for an IIR-type filter.

The Constant-Modulus Algorithm and the Property-Restoral Principle

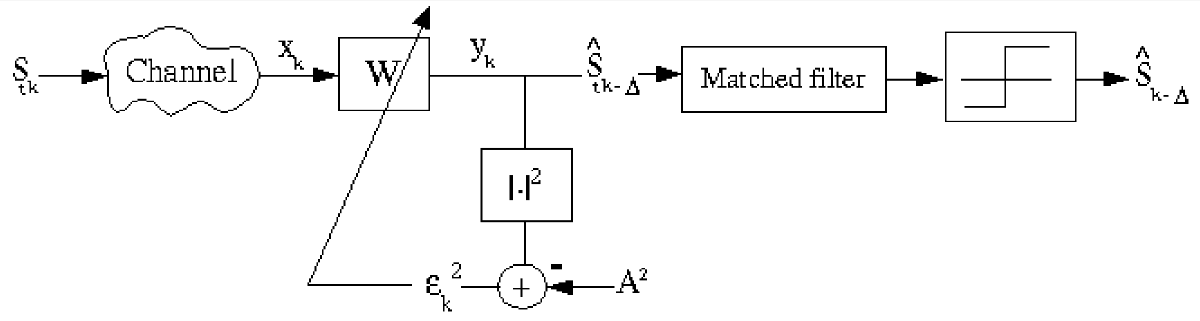
The adaptive filter configurations that we have examined so far require a "desired signal" d_k. There are many clever ways to obtain such a signal, but in some potential applications a desired signal is simply not available. However, a "property-restoral algorithm" can sometimes circumvent this problem.

If the uncorrupted signal has special properties that are characteristic of the signal and not of the distortion or interference, an algorithm can be constructed which attempts to cause the output of the adaptive filter to exhibit that property. Hopefully, the adapting filter will restore that property by removing the distortion or interference!

Example:
the Constant-Modulus Algorithm (CMA)
Certain communication modulation schemes, such as PSK and FSK, transmit a sinusoid of a constant analytic magnitude. Only the frequency or phase change with time. The constant modulus algorithm tries to drive the output signal to one having a constant amplitude:

$$\varepsilon_k = (|y_k|)^2 - A^2$$

One can derive an LMS (or other) algorithm that seeks a Wiener filter minimizing this error. In practice, this works very well for equalization of PSK and FSK communication channels.



CMA is simpler than decision-directed feedback, and can work for high initial error rates!

This property-restoral idea can be used in any context in which a property-related error can be defined.

Complex LMS

LMS for complex data and coefficients (such as quadrature communication systems) takes the form

$$y_k \qquad W_k^H X_k$$

$$e_k \qquad d_k \qquad y_k$$

$$W_k \qquad\qquad W_k \qquad \mu e_k X_k$$

It is derived in exactly the same way as LMS, using the following complex vector differentiation formulas

$$\frac{}{} \quad P^H$$

$$\frac{}{} \quad W^H$$

$$\frac{}{} \quad W^H R \qquad\qquad R$$

or by differentiating with respect to the real and imaginary parts separately and recombining the results.

Normalized LMS

In "normalized" LMS, the gradient step factor μ is normalized by the energy of the data vector:

$$\mu_{\text{NLMS}} = \frac{\alpha}{X_k^H X_k + \sigma}$$

where α is usually $\frac{1}{2}$ and σ is a very small number introduced to prevent division by zero if $X_k^H X_k$ is very small.

$$W_{k+1} = W_k + \frac{1}{X^H} e_k X_k$$

The normalization has several interpretations

1. corresponds to the 2nd-order convergence bound
2. makes the algorithm independent of signal scalings
3. adjusts W_{k+1} to give zero error with current input: $W_{k+1} X_k = d_k$
4. minimizes mean effort at time $k + 1$

NLMS usually converges **much** more quickly than LMS at very little extra cost; NLMS is **very** commonly used. In some applications, normalization is so universal that "we use the LMS algorithm" implies normalization as well.

Summary of Adaptive Filtering Methods

1. **LMS** remains the simplest and best algorithm when slow convergence is not a serious issue (typically used) $O(N)$
2. **NLMS** simple extension of the LMS with much faster convergence in many cases (very commonly used) $O(N)$
3. **Frequency-domain methods** offer computational savings ($O(\log N)$) for long filters and usually offer faster convergence, too (sometimes used; very commonly used when there are already FFTs in the system)
4. **Lattice methods** are stable and converge quickly, but cost substantially more than LMS and have higher residual EMSE than many methods (very occasionally used) $O(N)$
5. **RLS** algorithms that converge quickly and are stable exist. However, they are considerably more expensive than LMS. (almost never used) $O(N)$
6. **Block RLS** (least squares) methods exist and can be pretty efficient in some cases. (occasionally used) $O(\log N)$, $O(N)$, $O\ \ N^2$
7. **IIR** methods are difficult to implement successfully and pose certain difficulties, but are sometimes used in some applications, for example noise cancellation of low frequency noise (very occasionally used)
8. **CMA** very useful when applicable (blind equalization); CMA is **the** method for blind equalizer initialization (commonly used in a few specific equalization applications) $O(N)$

Note: In general, getting adaptive filters to work well in an application is much more challenging than, say, FFTs or IIR filters; they generally require lots of tweaking!